REMARKS

Claims 1-15 are pending in the application. Claims 1-15 are rejected. The specification is objected to. The drawings are objected to. All objections and rejections are respectfully traversed. Claims 1 and 8 are amended herein. Claim 12 is cancelled. No new subject matter is added.

In paragraph 1, the Examiner notes the claims have been examined after the entry of a Preliminary Amendment mailed on December 6, 2000.

In paragraph 2, the drawings are objected to according to an attached form PTO-948. A proposed Drawing Amendment is submitted herewith.

In paragraph 3, the Examiner objects to the specification because it is not clear to the Examiner whether the U.S. government retains license rights to the application. The Applicants assert that the U.S. Government does not retain rights to the application.

In paragraph 4, claims 1-15 are rejected under 35 U.S.C. 112, first paragraph, as containing subject matter which was not described in the specification in such a way as to enable one skilled in the art to which it pertains, or with which it is most nearly connected, to make or use the invention.

In particular, the Examiner asserts that the specification fails to provide any description regarding "the type of object-oriented programming structure used to model an object."

A person of ordinary skill in the art would readily understand that object oriented programming (OOP) ***is*** a type of programming structure.

To those of ordinary skill in the art, object oriented programming has been known for almost half a century. There are literally hundreds and hundreds of reference text books, and thousands technical papers that described object oriented programming to minute details. Object oriented programming is well known and a common technique in programming application and operating systems.

Certainly, the Examiner must be familiar with the seminal text on this subject by O.-J. Dahl, E. W. Dijkstra and C. A. R. Hoare, *Structured Programming*, Academic Press London and New Academic Press London and New York, 1972. York, 1972.

There are many many different number of languages associated with OOP, e.g., SmallTalk, Lisp, Cobra, SQL, Stimula, Java and C++, to name but a few, which are commonly used by programmers for object oriented programming.

In fact, Applicants are hard pressed to name any modern programming language that is ***not*** object oriented.

In object oriented programming, an object includes data and a method that operates on the data. That is the basic definition of an object. No further explanation is required. The Applicants are mystified that the Examiner is not familiar with object oriented programming, or does not understand this simple definition.

In other terms, the object contains an operand (data), and an operator (method, functions, etc.) that operates on the operand. In this way, the data structure becomes an *object* that includes both data and functions. In addition, programmers can create relationships between one object and another. For example, objects can *inherit* characteristics from other objects.

One of the principal advantages of object-oriented programming over procedural programming techniques is that it enables programmers to create modules that do not need to be changed when a new type of object is added. A programmer can simply create a new object that inherits many of its features from existing objects. This makes object-oriented programs easier to modify and maintain.

To perform object-oriented programming, one needs an *object-oriented programming language (OOPL).* Java, C++ and Smalltalk are three of the more popular languages, and there are also object-oriented versions of Pascal, which was developed in the 1960s. C++ and Java are the most common programming languages used nowadays. Every programmer knows that these are object oriented programming languages.

Object oriented programming is a well known programming structure, see, e.g., U.S. Patent 6,636,866 "System and method for object representation in an object-oriented programming language," U.S. Patent 6,694,506 "Object oriented programming system with objects for dynamically connecting functioning programming objects with objects for general purpose operations," U.S. Patent 6,694,507 "Method and apparatus for analyzing performance of object oriented programming code," U.S. Patent 6,298,476, "Object oriented software build framework mechanism," U.S. Patent 6,237,044 "Method for object-oriented programming using dynamic interfaces." A person of ordinary skill in the art of would readily understand how to generate a model as claimed using an object oriented programming structure.

Generating a model according to the invention is a simple exercise for a programmer, e.g., a Java programmer, of ordinary skill. Input data include coordinates expressed in Euclidean space for a plurality of points **x** for each component of an object. There is no mystery in that step for a Java programmer.

Next, for each component, each point **x** is encoded as a null vector x in a homogeneous space by $x = (\mathbf{x} + \frac{1}{2}\mathbf{x}^2 e + e_*)E = \mathbf{x}E - \frac{1}{2}\mathbf{x}^2 e + e_*$, where e and e_* are null vectors of with unit bivector $E = e \wedge e$. The above transform is explicitly recited in the specification and claim. The Examiner's rejection is not understood, because this transform requires no more than two or three lines of code to implement. To any person of ordinary skill of the art, this is a trivial exercise.

A plurality of general homogeneous operators are associated with each component to generate a model of the object. Again, the specification provides numerous, easily encoded operators for associating with the components.

The Examiner further asserts that the specification fails to provide any description of the type and values of "run-time parameters 121" and guidance describing how one of ordinary skill in the art would select such types and values for run-time parameters.

Types and values of run time parameters are user selected, depending on, e.g., user preference, the object being modeled, etc. It has been known for over thirty years, and would be readily apparent to a person of ordinary skill in the art how to supply run time parameters for the plurality operators as claimed. The types and values depend on the user, the object being modeled, etc.

Additionally, the Examiner asserts that the Specification lacks flowcharts or text describing the steps necessary to program an encoder to generate an object-oriented programming structure and to program a modeler to operate on the structure. As stated above, the actual programming of the encoder and modeler require only a normal level of knowledge for an ordinarily skilled programmer.

It is well known that an advantage of an OOP structure is that it does not require flow charts. Flow charts are archaic. This is well founded in Edsger Dijkstra's now famous 1968 letter entitled "GO TO Statement Considered Harmful," that launched the battle against flowcharts and spaghetti code, and introducing the idea of structured or object-oriented programming

Its is well known that flow charts have all kinds of problems. OOP does need to use flow charts, please see Dijkstra letter. The Examiner's request for a flowchart is a step back to the dark ages of programming.

MPEP 2106.01 states "when basing a rejection on the failure of the applicant's disclosure to meet the enablement provisions of the first paragraph of 35 U.S.C. 112, the examiner must establish on the record that he or she has a reasonable basis for questioning the adequacy of the disclosure to enable a person of ordinary skill in the art to make and use the claimed invention without resorting to *undue experimentation*." The Examiner has failed to provide any reasonable basis for questioning the adequacy of the specification. The Examiner bases his rejection on lack of adequate disclosure of actual programming steps. The specific acts of programming the operators and data of the invention are widely known. The Specification provides numerous explicit examples of operators and types and form of data representing components to enable any programmer familiar with object oriented programming to make of use the invention with little or no experimenting. Therefore, the applicants respectfully request the rejection be reconsidered and withdrawn.

Claims 1-15 are rejected under 35 U.S.C. 101 for reciting a process that is not directed to the technological arts.

Claims 1-15 are further rejected for reciting a process comprising an abstract idea. The examiner asserts that the recited steps do not recite data gathering limitations or post-mathematical operations that might independently limit the claims beyond performance of a mathematical operation, or limit the use of the output to a practical application providing a useful, concrete and tangible result.

The Applicants respectfully assert that the claims as amended apply directly to practical applications in the field of rigid body modeling.

Specifically, a method models a moving object composed of components. Input data specifies the components. The components are encoded and associated with a homogeneous space to generate a model of the object from which motion can be determined. Claims do not get any more real than that.

In the vernacular, the claims as amended address fundamental problems in rigid body modeling and programming. Claims 1 and 2 encode the position of material particles or points (components) in a rigid body. Claims 3 to 11 describe lines, planes, spheres and distances between them, as required in practical applications for modeling shape, separation and contact of rigid body parts. Claim 13 models rigid body dynamics, and how the model can be expressed in terms of standard variables, i.e., rotational and translational velocities. Claims 14 and 15 model a chain of linked rigid bodies, as required for practical applications to robotics.

These points about the applicability of the claims to fundamental problems in rigid body modeling will be understood to any person of ordinary skill in the art.

No prior art anticipating the invention is cited, and therefore, the claims are allowable.

In view of the foregoing, it is respectfully submitted that the application is in condition for allowance and an early indication of the same is courteously solicited. The Examiner is respectfully requested to contact the undersigned by telephone at the below listed telephone number, in order to expedite resolution of any remaining issues and further to expedite passage of the application to issue, if any further comments, questions or suggestions arise in connection with the application.

To the extent necessary, a petition for an extension of time under 37 C.F.R. 1.136 is hereby made. Please charge any shortage in fees due in connection with the filing of this paper, including extension of time fees, to Deposit Account 50-0749 and please credit any excess fees to such deposit account.

Respectfully submitted,
Mitsubishi Electric
Research Laboratories, Inc.

[illegible]

Andrew J. Curtin
Registration No. 48,485

201 Broadway, 8th Floor
Cambridge, MA 02139
Telephone: 617-621-7573
Facsimile: 617-621-7550